(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
29 November 2001 (29.11.2001)

PCT

(10) International Publication Number
WO 01/90903 A1

(51) International Patent Classification⁷: G06F 13/00

(21) International Application Number: PCT/US01/16658

(22) International Filing Date: 23 May 2001 (23.05.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/577,545 24 May 2000 (24.05.2000) US

(71) Applicant: COHERE NETWORKS, INC. [US/US]; Suite 1000, 6400 Fiddler's Green Circle, Englewood, CO 80111 (US).

(72) Inventors: MITCHELL, James, C.; 1181 Eagle Road, Broomfield, CO 80020 (US). RAMASWAMY, Arun; 6565 South Syracuse Way, Apt. 1010, Englewood, CO 80111 (US). BOSWORTH, Alan, N.; 15509 Eastbourn Drive, Odessa, FL 33556 (US).

(74) Agent: PINTO, James, A.; Dorsey & Whitney LLP, Republic Plaza Building, Suite 4700, 370 Seventeeth Street, Denver, CO 80202-5647 (US).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW.

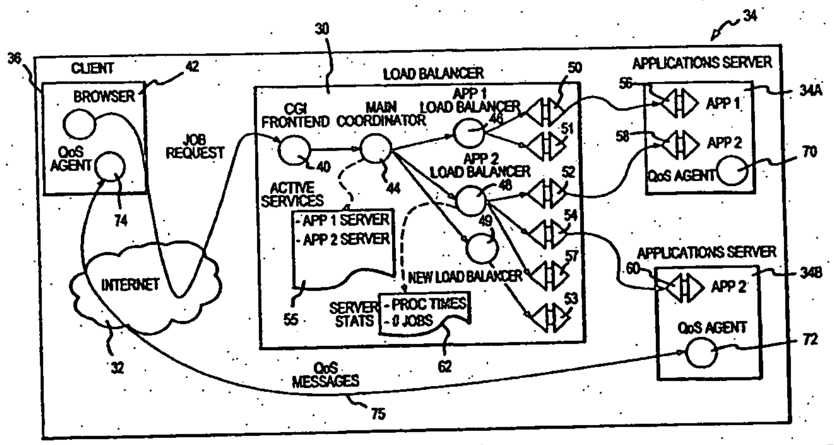
(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:
— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii)) for all designations*
— *as to the applicant's entitlement to claim the priority of the earlier application (Rule 4.17(iii)) for all designations*

Published:
— *with international search report*

[Continued on next page]

(54) Title: APPARATUS, SYSTEM, AND METHOD FOR BALANCING LOADS TO NETWORK SERVERS



(57) Abstract: A device (30), system, and method for determining if a request from a client computing station (36) for a service in a network should be processed by a first server (34a) or a second server (34b) adapted to service the request. The device includes a front end module (40) for receiving the request and translating the request into a transparent message format, a coordinating module (44) for determining if the first server and second servers are active, and at leat one load balancing module (46), in communications with the first and second servers, for determining whether the first server should service the request, and if so, passing the request to the first server. The load balancing module obtains various metics, such as quality of service (QOS) and number of pending request in the servers, from the servers from the server for determing if the first or second server should service the request. The device is also adapted to permit the dynamic addition of new servers to the device, or recognize the addition of new services provided by the servers, without having to alter or restart the device.

APPARATUS, SYSTEM, AND METHOD FOR BALANCING LOADS TO NETWORK SERVERS

FIELD OF THE INVENTION

The present invention relates, in general, to balancing requests for services, or loads, to network servers.

BACKGROUND OF THE INVENTION

In a client-server computing environment, such as the networked environment of the Internet, web sites offer a variety of services to the users (clients) via computer programs operating on one or more servers coupled to the network. In a simple server implementation of the web site, a single server hosts the various programs that form a web site, and as each request or "load" from a client is received at the server, the server performs the requested operation and passes data to the client, thereby satisfying the request (i.e., downloading text, audio, or video data to the client for display in the client's network browser program). In this simple model, difficulties can arise in servicing multiple requests from multiple clients for services from a single web site, as the server may not have the processing speed or throughput to service each of the multiple requests in a timely fashion.

One conventional approach to address this problem is shown in Fig. 1. Fig. 1 illustrates a client-server environment wherein a plurality of servers 20 is coupled to a network 22, such as the Internet, for providing various services from a single web site to one or more clients 24. A load balancing device 26 employing a conventional "round-robin" algorithm is provided between the servers 20 and the network 22. The servers 20 of the web site are configured as redundant servers, each having the same programs thereon to provide the same services from the web site to the clients 24. As requests for services are received at the web site, the load balancing device 26 passes each new request to the next server in a "round-robin" fashion. However, such an approach may still suffer from performance difficulties.

Accordingly, what is needed is an apparatus, system and method for balancing requests and loads from clients to servers of a web site in a computing network. It is against this background that various embodiments of the present invention were developed.

SUMMARY OF THE INVENTION

According to one broad aspect of the invention, disclosed herein is a device, also referred to herein as a load balancing device/apparatus, for determining if a request from a client computing station for a service in a network should be processed by a first server adapted to service the request or by a second server adapted to service the request. The device includes a front end module for receiving the request and translating the request into a transparent message format, a coordinating module for determining if the first server and second servers are active, and at least one load balancing module, in communications with the first and second servers, for determining whether the first or second server should service the request, and passing the request to the appropriate first or second server, as determined thereby.

In one example, the front end module translates the request into either an XML format (extensible markup language) or a binary format. Preferably, the load balancing module receives a quality of service metric or other data from the first server and from the second server, and determines whether the first or second server should service the request based in part on the metrics. As used herein, the term "quality of service" (QoS) includes, but is not limited to, one or more measures or metrics of the responsiveness of a server in satisfying a client's request for service over a network. QoS and the associated metrics provide information or data regarding the total network system response, and may be affected by, for example, sever loading, network loading, burst traffic, or the like.

Also, the load balancing module can obtain the number of pending requests at the first server, a number representing the time required to service the pending requests by the first server, the number of pending requests at the second server, and a number representing the time required to service the pending requests by the second server. In this example, the load balancing module determines whether the first server should service the request based, at least, on the quality of service metrics obtained from the first and second servers, the number of pending requests at the first server, the number representing the time required to service the pending requests at the first server, the number of pending requests at the second server, and the number representing the time required to service the pending requests at the second server.

The device can also include a first communications interface to the first server for coupling the load balancing module to the first server, a second communications interface to the second server for coupling the load balancing module to the second server, and a third communications interface reserved for the dynamic addition of a third server for coupling the load balancing module to the third server. Additionally, the device can include an additional load balancing module reserved for the dynamic addition of a new service.

According to another broad aspect of the invention, disclosed herein is a system for receiving and servicing a request from a client computing station for a service in a network. The system include a first server adapted to service the request, a second server adapted to service the request, and a device for determining if the request should be processed by the first server or the second server. Preferably, the device includes a front end module for receiving the request and translating the request into a transparent message format, a coordinating module for determining if the first server and second servers are active, and at least one load balancing module, in communications with the first and second servers, for determining whether the first server should service the request, and if so, passing the request to the first server.

Preferably, the first server and second server each have an input queue for tracking the pending requests to be processed by the first server, and each maintain a list of pending requests and a number corresponding to the time for completing each of the pending requests.

Further, the system preferably includes a quality of service agent operating at the client, a quality of service agent operating on the first server adapted to communicate with the quality of service agent operating at the client, and a quality of service agent operating on the second server also adapted to communicate with the quality of service agent operating at the client.

The quality of service agent operating on the first server is also adapted to communicate with the load balancing module with a message containing data of the quality of service between the client and the first server. Likewise, the quality of service agent operating on the second server is adapted to communicate with the load balancing module with a message containing data of the quality of service between the client and the second server. The load balancing module determines whether the first server should service the request based in part the data of the quality of service between the client and the first server and the data of the quality of service between the client and the second server.

According to another broad aspect of the invention, disclosed herein is a method for distributing a request from a client for a service from a web site having a plurality of servers adapted to service the request. The method includes receiving the request and determining if the service requested is offered by a first server and a second server of the plurality of servers. In one embodiment, a "quality of service" (QoS) metric is obtained from the first server, and a quality of service metric is obtained from the second server. Based, at least, on the quality of service metrics obtained from the first and second servers, a determination is made whether the first server should service the request, and if so, the request is passed to the first server.

The quality of service metric can takes the form of a measure of the performance being provided by a particular server to a client, for instance during the duration of the service period (i.e., during the transmission of data from the server to the client). In one example, a client agent operating on the client is provided, and a server agent operating on the first server is provided. In one example, the client agent transmit a message to the server agent, the message containing a data rate of data transferred from the client to the first server, wherein the data rate is used as a quality of service metric of the first server. This data is used to determine which server should service the request of the client.

In another embodiment, the number of pending requests at the first server is obtained, as is the time (estimated, actual, or empirical) required to service the pending requests by the first server. Similarly, the number of pending requests at the second server is obtained, along with the time required to service the pending requests by the second server. The determining step then determines whether the first server should service the request based, at least, on the quality of service metrics obtained from the first and second servers, the number of pending requests at the first server, the time required to service the pending requests at the first server, the number of pending requests at the second server, and the time required to service the pending requests at the second server. Again, the time required can be an estimated time, actual time, or empirical time.

The quality of service metrics, and other performance characteristics of the system, can be remotely accessed if desired.

In another embodiment, the client's request is translated into a transparent message format usable by the first and second server, such as XML format with a start designator identifying the beginning of the message, a type designator identifying the type of service, and an end designator indicating the end of the message. Binary format can also be used.

Furthermore, the method of the present invention permits dynamic additions of additional servers to the system. Upon an addition of a third new server to the web site, the presence of the new third server is detected and it is determined if the new third server offers the service requested by the client. A quality of service metric is obtained from the new third server and is included in the determination of whether the first server should service the request.

Moreover, the method of the present invention permits dynamic addition of a new service on either an existing server or a new server to the system. Upon addition of a new service to the web site, the presence of the new service is detected whereupon the load balancer offers the service requested by a client. A quality of service metric is obtained from the server managing the new service and is included in the determination of whether the server offering the new service should receive and process the client request.

The foregoing and other features, utilities and advantages of the invention will be apparent from the following more particular description of a preferred embodiment of the invention as illustrated in the accompanying drawings and claims.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 illustrates a block diagram of conventional client-server system having a load balancing device utilizing a conventional "round-robin" algorithm for balancing loads in a network such as the Internet.

Fig. 2 illustrates a block diagram of one embodiment of the present invention.

Fig. 3 illustrates a distribution of requests/loads internal to a server in accordance with one embodiment of the present invention.

Fig. 4 illustrates an example of the logical operations performed by the load balancer in accordance with one embodiment of the present invention.

Fig. 5 illustrates an example of the logical operations performed by a server in accordance with one embodiment of the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

In accordance with the present invention, a load balancing apparatus and method therefor, as well as a system using the same, is disclosed herein. In particular, the load balancing apparatus, referred to variously herein as a "load balancer" or a "load balancing device" employs a unique and novel set of decision criteria in determining which server coupled thereto should receive and process a request or "load" from a client over the network.

Referring now to Fig. 2, a load balancer 30 in accordance with one embodiment of the present invention is shown. The balancer 30 is an interface between the network 32 and a plurality of servers 34 (two applications servers 34A, 34B are shown in the example of Fig. 2). Each server 34A,B has a set of software programs, such as "App1" and "App2" shown in Fig. 2, for providing the services offered by the website as requested by one or more clients 36 in the network. Each server 34A,B also has an input queue therein, as will be described later with reference to Fig. 3.

In particular, server 34A provides service "App1" and service "App2", while server 34B provides service "App2." As will be discussed in greater detail below, in response to a request from client 36 for service "App2", the load balancer 30 of the present invention determines whether server 34A or server 34B should process the client's request for service "App2", and upon such determination, the load balancer passes the request for service "App2" to the selected server.

In accordance with the present invention, the load balancer 30 shown in Fig. 2 has knowledge of which applications ("App1, "App2") are loaded on which servers 34A,B, so that the load balancer 30 can pass the client's 36 request for a particular service to the appropriate server or set of servers. Furthermore, the load balancer 30 has knowledge of various server specific "metrics" which are criteria used by the load balancer 30 to determine which server should service a pending request from a client 36. In one example, the load balancer 30 of the present invention receives information from each server 34A,B relating to the number of pending jobs in the input queue of that server, as well as the time required for each job to be completed by that server. Further, the information from the server 34A,B may include percent utilization of CPU cycles, percent utilization of network, and other metrics that the server can collect from its own operating system registry. In this manner, the load balancer 30 can compute a numerical metric which is the product of the number of pending jobs in the input queue of a server, multiplied by the time required to complete each job. Accordingly, the load balancer 30 then has information relating to the ability of a particular server to process an incoming request or load from a client 36. In another example, the load balancer also receives a "quality of service" value from monitoring processes running on the client and server platforms, described below.

For example, if the number of jobs pending in the queue of server 34B is one, and the time required to complete a job is approximately 1.5 seconds, then in this example the metric would be 1.5. In contrast, if server 34A had three jobs pending in its queue and required 1.0 second to complete a job, then the metric for server 34A would be three. Accordingly, for an incoming request for services or load from a client 36, the load balancer 30 would pass the next incoming request to server 34B, as its metric indicates that server 34B is more available to handle the incoming request than is server 34A.

The load balancer 30 as shown in Fig. 2 has a number of processes and interfaces in accordance with one embodiment of the present invention. A CGI front-end process 40 is provided for receiving data from a client application or network browser 42, and converting the data into a desired format. The CGI front-end process 40 is associated with one or more services provided by application servers 34A,B, assuming that the application servers 34A,B have previously "registered" with the load balancer (registration is described below). In one example, the incoming data is converted by the CGI front end process 40 from a plurality of client formats, into a format compatible with the requested application and compatible with the remaining load balancing processes. In one example, the CGI front-end process 40 converts the message format into a "transparent messaging" format usable by the load balancing processes. Transparent messaging enables the various internal processes of the load balancer to route and load balance network requests without knowing the content of each message itself. In this way, the message content is transparent to the load balancing system.

The format of a transparent message is an encapsulation in which the application specific data is encapsulated in the payload or central portion of the message. Around the payload is a start and type designator in the front portion of the message, and a stop designator/identifier at the back portion of the message. Two embodiments of the transparent messages are shown in Table 1 and 2. The first embodiment is in a binary format for efficiency and compatibility with binary data. The second embodiment uses the industry standard XML (extensible markup language) ASCII-based notations for use with strictly ASCII messages and for extensibility of future applications.

Table 1. Format example of binary based transparent messages.

```
Byte:  0    4 5            6...                          N   N+4
Data:  START <type byte> <variable number of bytes> STOP
```

Table 2. Format example of XML based transparent messages.

```
<?xml version="1.0"?>
<MessageContent>
  <ServiceName>type designator</ServiceName>
       Application Data …
</MessageContent>
```

In the binary format shown in Table 1, the start designator is the hexadecimal equivalent of the character string value "START." The stop designator is the hexadecimal equivalent value of the character string value "STOP." In one example, the type byte is an 8 bit value that is registered with CGI front-end process 40 and represents the type of application service requested, corresponding for example to "Appl" or "App2" shown in Fig. 2. The remaining message content is formatted for the specific application service being requested. Using this technique, the load balancer 30 does not need to understand the content of the message to be able to forward the message to a compatible application server 34A,B that is available and which can most efficiently process the load.

In the XML format shown in Table 2, the start designator is the XML compatible string "<ServiceName>". The end designator is the XML compatible string "</ServiceName>". The type designator is an XML compatible string located between the start and stop designators without extra spaces. In this format, the start and stop designators identify the location of the type designator within a compatible XML message. Using this technique, the load balancer 30 does not need to understand the content of the message in order to be able to forward the message to a compatible application server 34A,B.

The use of XML compatible message formatting provides an extensible and simple method of adding new services to the load balancing system without requiring changes to the load balancing algorithms, software or processes employed. Further, providing transparent messaging results in greater speed and efficiency and eliminates any need for re-compiling due to changes in the content of application's messages.

After formatting by the CGI front-end 40, the transparent message is passed to the Main Coordinator process 44, referred to also herein as a "coordinator module," of the load balancer 30 for decoding and forwarding to the appropriate load balancing module/process 46, 48, and ultimately to the appropriate selected server through the appropriate communications thread 50, 52, or 54.

The Main Coordinator process 44 decodes the type designator to identify to which load balancing module 46, 48 the message should be forwarded. The Main Coordinator process 44 maintains a list 55 of the load balancing modules 46, 48 and their associated applications (i.e., "App1", "App2"). If the client's 36 request refers to a service that is to be provided by one of the plurality of servers 34A,B coupled to the load balancer 30, then the Main Coordinator process 44 passes the request to the appropriate load balancing module 46, 48 for further processing. Otherwise, if the Main Coordinator process 44 determines that the client's 36 request is not addressed to one of the plurality of servers 34A,B coupled to the load balancer 30, in one example, the Main Coordinator process 44 replies to the client's request with a "service not available" message.

The load balancing modules 46, 48 determine which server should service the client's request based on various metrics relating to each server 34A,B, such as quality of service, number of pending jobs, or other decision criteria discussed herein or with respect to Fig. 4, or any combination thereof. For example, the load balancing module 48 of Fig. 2 determines whether a request for "App2" service should be processed by server 34A or server 34B. Upon determining which server should process the service request, the load balancing process 48 forwards the request to the chosen server.

The load balancing modules 46, 48 are coupled to each server 34A,B through a plurality of communication interfaces, shown as threads 50, 52, 54, with corresponding threads 56, 58, 60 at the servers 34A,B.

As previously described, one example of the metrics includes a calculation of the number of pending jobs in a particular server's input queue multiplied by the time required by the server to complete each job, shown as "Server Stats" 62 in Fig. 2. Alternatively, the load balancing decision process can account for a quality of service (QoS) figure, described below, in making its determination. Upon determining to which server 34A,B the client's request should be passed, the load balancing module 46, 48 then forwards the client's request through the proper communication interface to the appropriate server. The server 34A,B then places the request in its input queue as described below with reference to Fig. 3.

It will be understood that while the load balancing decision process is described as a portion of the functionality of the load balancing module implemented in various processes 40, 44, 46,48, such functionality can be combined or subdivided or otherwise arranged differently, and may reside at a portal or the like, and incorporated therein.

Further in accordance with one embodiment of the present invention, the quality of service "QoS" figure is provided and tracked throughout the system and provides valuable information to the load balancer 30 in making its determination as to which server 34A,B should process a client's request. In one example, and as shown in Fig. 2, quality of service agents 70, 72 are operated on each of the plurality of application servers 34A,B and quality of services agents 74 are operated on each of the client 36 platforms, and communicate QoS messages such as message 75 shown in Fig. 2. The QoS agents on the application server and the client communicate to each other over the duration of the provided service. Each agent sends QoS messages to the other respective agent, essentially reflecting back to the sending side what the receiving side is seeing in terms of network performance. In one example, the messages between the respective QoS agents contain a QoS agent identification number, sequence counter, time stamp, and other status information such as CPU percent utilization, average data rate, bits transferred, etc. The QoS agent 74 operated by the client communicates with the respective QoS agent 70, 72 operated by the server via these status messages, intrinsically measuring the quality of the network path there between. The QoS agent 70, 72 operated by the application server 34A,B supplies its performance metrics back to the load balancer 30 via a QoS message, which is application server dependent. This QoS message is fed back to the load balancer 30 at the beginning of each new request/load, or more often if desired. Each application server's 34A,B QoS messages are used by the load 30 balancer in its decision process of determining the best or most appropriate application server to handle a new request or load. By sending the QoS messages at a regular rate or "ping", the instantaneous and average network performance can be gauged by computing the latency of the messages as well as the variance in message latency. Furthermore, the status information provides a measure of the loads at various points in the networked system, from the clients to the application servers.

For instance, if a client user 36 was receiving a very good response time for file downloads, then the status information received from the client pings or messages would show a greatly increasing number of bytes transferred and a high average data rate. This would indicate that the respective application server 34A or 34B was performing well. However, the variance of ping latencies could be high indicating a large amount of burst data on the associated communications path. In the case of a file download, the load balancer 30 may ignore the latency variance due to the inherent variable data rate nature of a file download. In contrast, services such as streaming voice or video would be very sensitive to latency variation. In this case, a large variance in latency would, for example, trigger the load balancer 30 to reroute future requests to other servers, possibly using alternative communications paths.

Metrics computed from the data and status values in the QoS messages can be used in place of or in combination with the queue metric described above. For instance, average data rate of a server can be divided by the CPU percent utilization. This would yield a metric indicating the performance of the server, by which the plurality of servers 34A,B could be ranked by this metric. In this example, a new client request would be routed to the server with the highest ranking. Alternatively, the described metric could be mathematically divided by the variance of the latency calculated from the ping rate variance. In this approach, a high variance would reduce the ranking of a server, with a high variance resulting in a new distribution of message routing. In this sense, the QoS agents 70, 72 and 74 provide feedback to the load balancer 30 as to how well the services are sent by the respective server 34A,B and being received by the end user at the client station 36.

Referring to Fig. 3, and in accordance with the present invention, a server implementation is shown for a server, such as server 34A of Fig. 2, coupled to the load balancer 30 of the present invention. The server 34A has an input queue 80 for storing requests for services, and a job statistics table 82 for storing data relating to the server metrics. The input queue 80 can be implemented as a global queue for all incoming requests, or as a set of local input queues, each associated with a particular application (such as "App 1" or "App 2") provided by the server 34A. For the "App 1" application serviced by the server 34A, the server has a front end process/thread 84 and a plurality of processes 86A,B,C for servicing the requests placed in the respective locations of the queue. Similarly for the "App 2" application serviced by server 34A, a front end process/thread 88 and process 90 is provided.

Fig. 3 will be described with respect to a request for "App 1" service through front-end process 84. As a request from the load balancer is received, the appropriate front end thread/process 84 receives the request from the load balancer and places the request in the input queue 80. In one example, the input queue 80 is a circular queue having N entries, such that the front end thread/process 84 places an incoming request into the next available location in the input queue 80. If the input queue 80 is full, then the front end thread 84 of the server communicates to the load balancer, as part of the server metrics, that the input queue 80 is "full." In response, the load balancer avoids passing any further request to the particular server with the full input queue until the load balancer receives a subsequent message that the input queue 80 of the server is again available to accept and process new requests.

The "worker processes" 86A,B,C illustrated in Fig. 3 receive tasks to perform from an entry on the input queue 80. Each of these processes 86A,B,C is an executable image providing one of the services that may be requested by a client. When a process 86A,B,C has completed a requested service, it enters an idle state where it waits and periodically checks the input queue for a new service request. If there is a service request in the queue (*i.e.*, the queue is not empty), the process 86A,B,C copies any user parameters in the queue entry from the client user, deletes the queue entry, and begins performing the service requested. The deletion of the queue entry indicates that the slot is available for scheduling or queuing a new entry by the front-end process 84. After completing the requested service, the process 86A,B,C goes back into idle mode to look for a new entry in queue 80.

The general logic flow of the load balancing process and the queuing method is shown in Figs. 4 and 5 respectively. Referring to Fig. 4, the logical operations performed in one embodiment of the load balancer 30 are illustrated. These operations can be performed, for example, by the Main Coordinator process 44, or preferably by the load balancing modules 46, 48 shown in Fig. 2. Operation 100 is the idle state of the load balancer, wherein the load balancer waits for a message to be received or operation to be performed. Upon receiving an incoming message, the load balancer determines whether the message is a quality of service (QoS) message in operation 102. If so, then at operation 104, the load balancer calculates the particular metric being used for the balancing comparison. As described above, one embodiment of the metric is the average duration time for a job multiplied by the number of jobs in the server input queue. Other metrics as previously described can also be calculated by operation 104. In one example, an array is maintained which includes therein a dynamic list of servers, arranged by their availability. The calculated metric is used to sort the array of servers in operation 106 to select which server to send the next service request to. After completion, operation 106 returns control to the idle state to await a new message.

If, in operation 102, the received message is not a QoS message, then operation 108 determines whether the received message is a server status message. As previously described, the server status message contains, among other statistics, whether the server queue is full or not full indicating whether the server is available or not available respectively. If the received message is a server status message, then operation 110 determines whether or not the server is indicating that its input queue is full. If the queue is full, then operation 112, marks a metric array slot associated with the server as unavailable. If the queue is not full, then operation 114, marks the metric array slot associated with the server as available. After completion of operations 112 or 114, control is passed to the idle state 100.

If in operation 108, the received message is not a server status message, then operation 116 determines whether or not it is a request for service message. If it is not a request for service message, then the message is discarded and control is passed to idle operation 110. If the received message is a request for service message, then operation 118 determines if there is at least one server available for providing the service. If a server is not available, then operation 120 notifies the originating user that the service is unavailable and suggests that the user try again later. If a server is available, then operation 122 sends the request to the server with the least load, indicated by being at the top of the sorted metric array described with reference to operation 106. After completion of operations 120, 122, control is passed to the idle state 100.

In Fig. 5, one example of the logical operations associated with the application server input queue 80 of Fig. 3 is illustrated. Preferably, the front-end process 84, 88 of Fig. 3 performs these queue processing operations. Operation 130 is the idle state of the front-end process, where the front-end waits for a message or operation to be performed. Upon receiving an incoming message, the front-end determines whether it is a request for service message in operation 132. If it is not a request for service message, then the message is discarded and the process returns to the idle state in operation 130. Operation 134 determines if there is room in the input queue for a new service request. If there is room, an estimate of the time to complete the requested service is made in operation 136. The running average of request service times is calculated in operation 138. This running average can be calculated by at least two methods. First, for a batch service request, such as a streaming video broadcast, the average is calculated as the total amount of time required to process all requests at the server, divided by the total number of requests pending. Second, for an interactive service request, such as a software service, the average is calculated as the total time of some previous number of completed requests, divided by the number of previous requests. The service request and the time estimated are stored in the next open queue slot in operation 140. The processing loop is finished by decrementing the count of available queue slots in operation 142, sending the time statistics (including the estimated, average times or empirically derived time for completion) to the load balancer in operation 146, sending the number of pending requests for service in operation 146, and returning to idle state 130 and the next message. In this manner, the server has communicated to the load balancer the respective metrics for the server, in accordance with the present invention.

If the queue is found to be full in operation 134, then the load balancer is notified of a full queue in operation 148. As described above, the load balancer will suspend sending any messages to this server until the queue opens up. In this example, the front-end process waits a programmed time in operation 150 before checking the queue again in operation 152. The front-end process loops between operation 150 and 152 until a slot in the queue becomes available. When a slot becomes available, a message indicating that the queue is available is sent to the load balancer in operation 154. The front-end process then returns to the idle state 130 until another message is received.

Furthermore, in accordance with the present invention, the load balancer provides a remote monitoring capability. In one example, each server communicates the average time to service requests and the number of pending jobs to the load balancer. In effect, this operation concentrates the server load metrics for the entire network at the load balancer. A remote "dial-in" process could gather the load metrics from one or more of the load balancers in a network to obtain a global view on the performance and load on the entire network.

Furthermore, in accordance with the present invention, the load balancer is adapted to recognize, on a dynamic basis, the addition of a new server or the replacement of an existing server. The discovery, identification and coordination of the server pool are performed through a dynamic communications system. In general, the load balancer initiates or offers one additional communications channel at all times, shown for example in Fig. 2 as 51 or 57. In one example, when a new server is attached to the network, the new server makes a request to send a message to the load balancer and as a result, finds or discovers the additional channel of a load balancer. This allows the new server to uniquely identify itself to the load balancer and coordinate communications. The load balancer, in response to a message over the additional channel, gathers the new server's information and adds a new slot in the server statistics table. After the new server has been recorded by the load balancer, a new additional channel is opened and maintained until the server expressly terminates communications with the load balancer, or is otherwise determined to be absent.

Furthermore, in accordance with the present invention, the load balancer 30 is adapted to recognize, on a dynamic basis, the addition of a new service in association with either an existing server or a new server. The new service can be, for example, a new capability, function or utility performed by a server. The discovery, identification and coordination of the new service are performed through a dynamic communication service similar to the aforementioned dynamic server coordination.

In general and referring to Fig. 2, the Main Coordinator process 44 initiates one additional, generic load balancing process/module 49 to discover and manage a new service. The generic load balancing module 49 initiates or offers a generic communication channel 53. In one example, when a new service is attached to the network, the server managing this new service makes a request to send a message to the generic load balancing module 49 and as a result, finds or discovers the communication channel 53 of the generic load balancing module 49. A generic naming practice is used to facilitate the service to discover the available channel of the generic load balancing module. Once a new service has been associated with the generic load balancing module 49, the generic load balancing module 49 registers a new service name with the Main Coordination process 44 (for example, by using list 55), changes its name and channel name to reflect the new service, and the Main Coordination process 44 initiates yet another a new generic load balancing module (not shown) to replace the recently renamed load balancing module 49 in order to support the dynamic addition of yet another service.

Generally, the generic load balancing process/module 49 dynamically discovers any new servers and operates similar to load balancing modules/processes 46, 48 once it has been renamed. As with load balancing modules 46, 48, the newly named load balancing module 49 preferably uses QoS metrics to decide to which server to send service requests. The name of the load balancing module 49 registered with the Main Coordinator process 44 can then be used by client applications to request the new service offered thereby. The capability of dynamically adding new services results, in part, from the transparent messaging and QoS metric of embodiments of the present invention.

In one embodiment, named pipes are used for communications between the load balancer and the servers. Alternatively, sockets can be used. In either case, a naming convention can be used to assist the server in opening a communications channel and find the additional channel associated with the load balancer. In one example, the pipe or socket channel will be named after the service that is being load balanced. For example, TRIMEDIT_SOCKET could be used for the Trim Edit function in video content creation. In another example, GENERIC_SERVICE_SOCKET could be used for the generic load balancing process/module 49 (shown in Fig. 2) to facilitate the discovery and dynamic recognition of new services.

Referring to Fig. 2, the generic load balancing process/module 49 has initiated a new named pipe 53 offered to new services. If a new service were to be connected to the generic load balancing module 49, the new service would make an open call in software to the generic named pipe 53 resulting in a connection to the generic load balancing module 49. This action would initiate the registering of a new service, the load balancing module 49 would begin accepting requests for the new service, and the Main Coordinator process 44 would initiate yet another new generic load balancing module (not shown) to provide for yet another new service.

Referring to Fig. 2, the "App 2" load balancing process/module 48 is currently managing two servers 34A,B. In this example, in order to support the dynamic addition of a new third server, the load balancing process 48 would initiate/maintain a third named pipe 57. If a new server were to be connected to the load balancer, the new server would make an open call corresponding to "App 2" in software, resulting in the connection to this third named pipe 57. This action would add the new server to the server pool and the load balancer would begin to accept and pass service request messages to the new server. Upon completion, the load balancer would initiate/maintain yet another new additional named pipe (i.e., forth named pipe, not shown) to provide for the dynamic addition of another (i.e., fourth) new server.

Hence, embodiments of the present invention permit the dynamic addition of new servers to the load balancer 30, or recognize the addition of new services provided by the servers, without having to alter or restart the load balancer 30.

This same mechanism can be used to dynamically detect the removal of a server. When a server catastrophically goes down or is shut down gracefully, the application server end of the named pipe or socket closes. This is detected by the load balancer and indicates that the server is now unavailable. The load balancer can remove the server from its list and remove the named pipe or use the available channel for the new additional channel.

The invention can be embodied in a computer program product. It will be understood that the computer program product of the present invention preferably is created in a computer usable medium, having computer readable code embodied therein. The computer usable medium preferably contains a number of computer readable program code devices configured to cause a computer to affect the various functions required to carry out the invention, as herein described.

While the embodiments of the invention have been described with respect to Figs. 2-5 wherein a single client 36 is shown communicating with the load balancer 30 coupled to a pair of servers 34A,B, wherein server 34A offers services "App 1" and "App 2" and server 34B offers service "App 2," it will be understood that the present invention will be applicable to various computing configurations where the number of clients, load balancers, servers, and services will vary as a matter of choice depending on the particular implementation.

The embodiments of the invention described herein are preferably implemented as logical operations in a computing system. The logical operations of the present invention are implemented (1) as a sequence of computing implemented steps running on the computing system, or (2) as interconnected modules within the computing system. The implementation is a matter of choice dependent on the performance requirements of the computing system implementing the invention. Accordingly, the logical operations making up the embodiments of the invention described herein are referred to variously as operations, steps, or modules.

While the method disclosed herein has been described and shown with reference to particular steps performed in a particular order, it will be understood that these steps may be combined, sub-divided, or re-ordered to form an equivalent method without departing from the teachings of the present invention. Accordingly, unless specifically indicated herein, the order and grouping of the steps is not a limitation of the present invention.

The foregoing embodiments and examples are to be considered illustrative, rather than restrictive of the invention, and those modifications, which come within the meaning and range of equivalence of the claims, are to be included therein. While the invention has been particularly shown and described with reference to a preferred embodiment thereof, it will be understood by those skilled in the art that various other changes in the form and details may be made without departing from the spirit and scope of the invention.

We claim:

1. In a computer network, a method for distributing a request from a client computing station for a service from a web site having a plurality of servers adapted to service the request, comprising:

receiving the request;

determining if the service requested is offered by a first server and a second server of said plurality of servers;

obtaining a quality of service metric from said first server;

obtaining a quality of service metric from said second server;

based, at least, on the quality of service metrics obtained from the first and second servers, determining whether the first server should service said request; and

if the determining step determines that the first server should service the request, passing the request to the first server.

2. The method of claim 1, further comprising:

obtaining a number of pending requests at the first server, and obtaining a number representing the time required to service said pending requests by said first server; and

obtaining a number of pending requests at the second server, and obtaining a number representing the time required to service said pending requests by said second server;

wherein said determining steps determines whether the first server should service said request based, at least, on the quality of service metrics obtained from the first and second servers, the number of pending requests at the first server, the number representing the time required to service said pending requests at the first server, the number of pending requests at the second server, and the number representing the time required to service said pending requests at the second server.

3. The method of claim 1, further comprising:

providing a client agent operating on the client computing station;

providing a server agent operating on the first server; and

transmitting a message from the client agent to the server agent, said message containing a data rate of data transferred from the client computing station to the first server, wherein said data rate is used as a quality of service metric of said first server.

4. The method of claim 1, further comprising:

translating said request into a transparent message format usable by said first and second server.

5. The method of claim 1, further comprising:

translating said request into a transparent message format usable by said first and second server, wherein said translation uses XML format with a start designator identifying the beginning of the message, a type designator identifying the type of service, and an end designator indicating the end of the message.

6. The method of claim 1, further comprising:

receiving over the network a remote request for said quality of service metrics as a measure of network performance; and

communicating said quality of service metrics in response to said remote request.

7. The method of claim 1, further comprising:

upon an addition of a third new server to the web site, detecting the presence of the new third server;

determining if the new third server offers the service requested by the client;

obtaining a quality of service metric from the new third server; and

including the quality of service metric from the new third server in the determination of whether the first server should service the request.

8. The method of claim 1, further comprising:

upon the addition of a new service on one of said plurality of servers of the web site, detecting the presence of the new service;

determining if the new service is the same type of service requested by the client;

obtaining a quality of service metric from the server associated with the new service; and

including the quality of service metric from the server associated with the new service in determining whether the first server should service the request.

9. A device for determining if a request from a client computing station for a service in a network should be processed by a first server adapted to service the request or by a second server adapted to service the request, comprising:

a front end module for receiving the request and translating the request into a transparent message format;

a coordinating module for determining if the first server and second servers are active; and

at least one load balancing module, in communications with said first and second servers, for determining whether the first server should service said request, and if so, passing the request to the first server.

10. The device of claim 9, wherein said front end module translates said request into an XML format.

11. The device of claim 9, wherein said front end module translates said request into a binary format.

12. The device of claim 9, wherein said coordinating module determines if said request should be passed to the load balancing module.

13. The device of claim 9, wherein said load balancing module receives a quality of service metric from said first server and from said second server, and determines whether the first server should service the request based in part on said metrics.

14. The device of claim 9, wherein said load balancing module obtains a number of pending requests at the first server, obtains a number representing the time required to service said pending requests by said first server, obtains a number of pending requests at the second server, and obtains a number representing the time required to service said pending requests by said second server;

wherein said load balancing module determines whether the first server should service said request based, at least, on the quality of service metrics obtained from the first and second servers, the number of pending requests at the first server, the number representing the time required to service said pending requests at the first server, the number of pending requests at the second server, and the number representing the time required to service said pending requests at the second server.

15. The device of claim 9, further comprising:

a first communications interface to said first server for coupling said load balancing module to said first server;

a second communications interface to said second server for coupling said load balancing module to said second server; and

a third communications interface reserved for the dynamic addition of a third server for coupling said load balancing module to said third server.

16. The device of claim 9, further comprising:

a second load balancing module reserved for dynamically recognizing a second service.

17. A system for receiving and servicing a request from a client computing station for a service in a network, comprising:

a first server adapted to service the request;

a second server adapted to service the request; and

a device for determining if the request should be processed by the first server or the second server, said device comprising:

a front end module for receiving the request and translating the request into a transparent message format;

a coordinating module for determining if the first server and second servers are active; and

at least one load balancing module, in communications with said first and second servers, for determining whether the first server should service said request, and if so, passing the request to the first server.

18. The system of claim 17, wherein said first server has an input queue for tracking the pending requests to be processed by the first server.

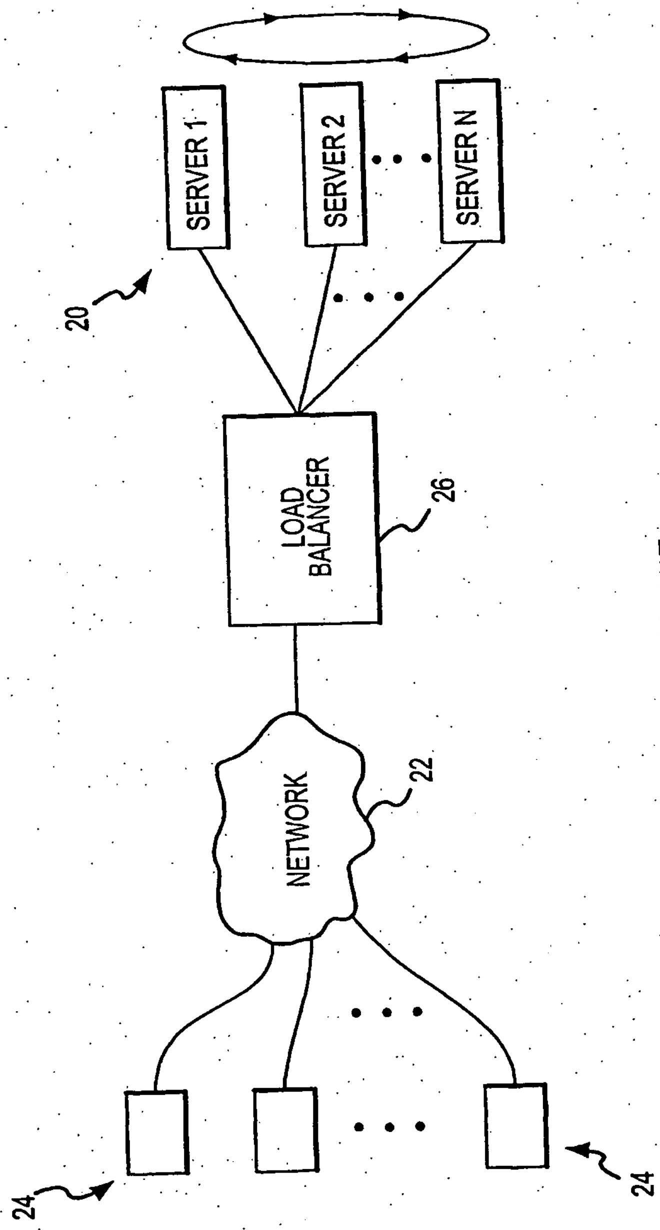
19. The system of claim 17, wherein said first server maintains a list of pending requests and a number corresponding to the time for completing each of the pending requests.

20. The system of claim 17, further comprising a quality of service agent operating at the client;

a quality of service agent operating on said first server adapted to communicate with the quality of service agent operating at the client, and adapted to communicate with said load balancing module with a message containing data of the quality of service between the client and the first server; and

a quality of service agent operating on said second server adapted to communicate with the quality of service agent operating at the client, and adapted to communicate with said load balancing module with a message containing data of the quality of service between the client and the second server.

21. The system of claim 20, wherein said load balancing module determines whether the first server should service said request based in part the data of the quality of service between the client and the first server and the data of the quality of service between the client and the second server.



PRIOR ART
FIG. 1

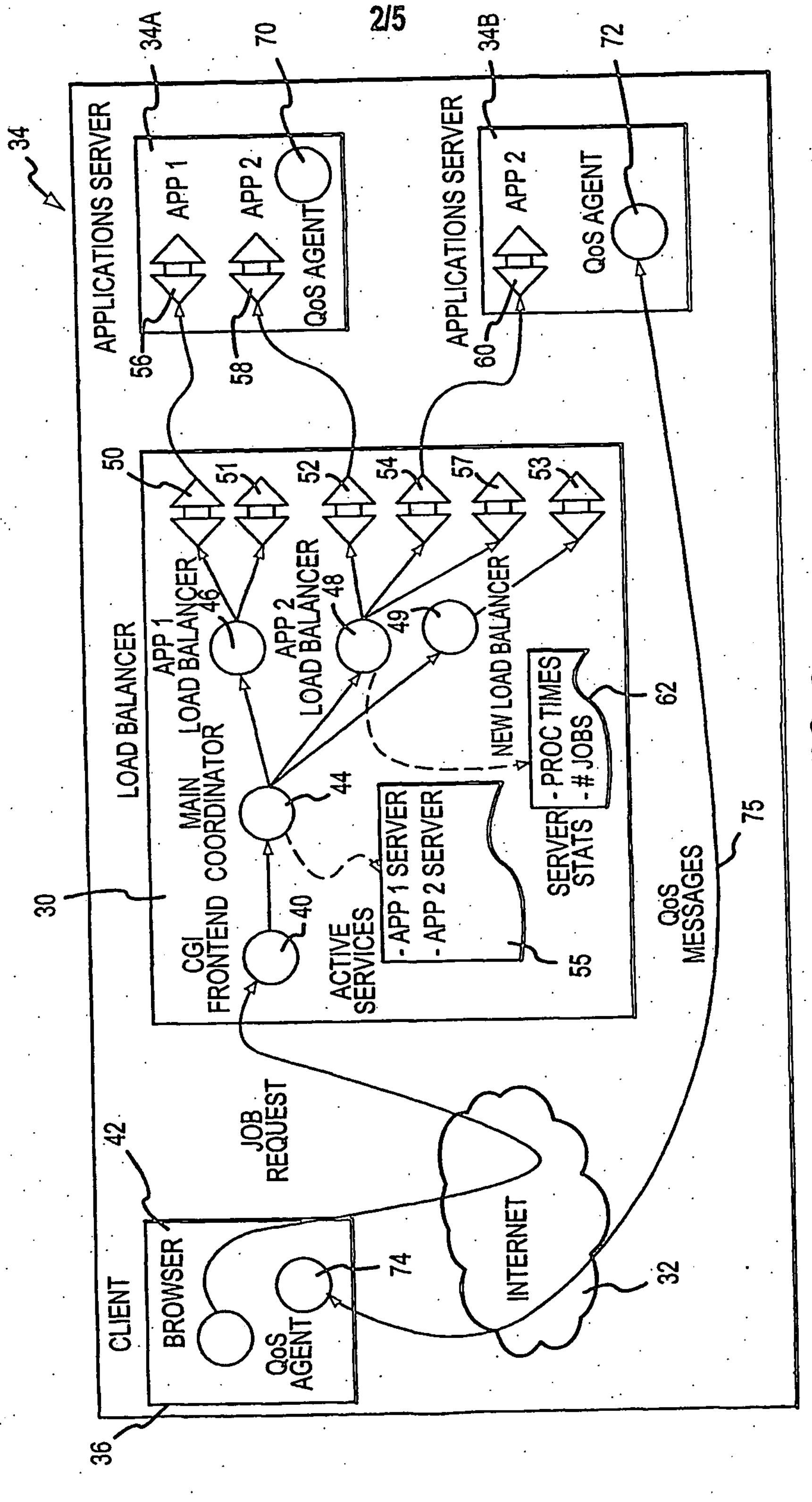

34
APPLICATIONS SERVER
34A
APP 1
APP 2
QoS AGENT
70
56
58
APPLICATIONS SERVER
34B
APP 2
QoS AGENT
72
60
LOAD BALANCER
30
CGI FRONTEND
40
MAIN COORDINATOR
44
APP 1 LOAD BALANCER
46
APP 2 LOAD BALANCER
48
49
NEW LOAD BALANCER
50
51
52
54
57
53
ACTIVE SERVICES
- APP 1 SERVER
- APP 2 SERVER
55
SERVER STATS
- PROC TIMES
- # JOBS
62
QoS MESSAGES
75
JOB REQUEST
CLIENT
BROWSER
42
QoS AGENT
74
INTERNET
32
36

FIG.2

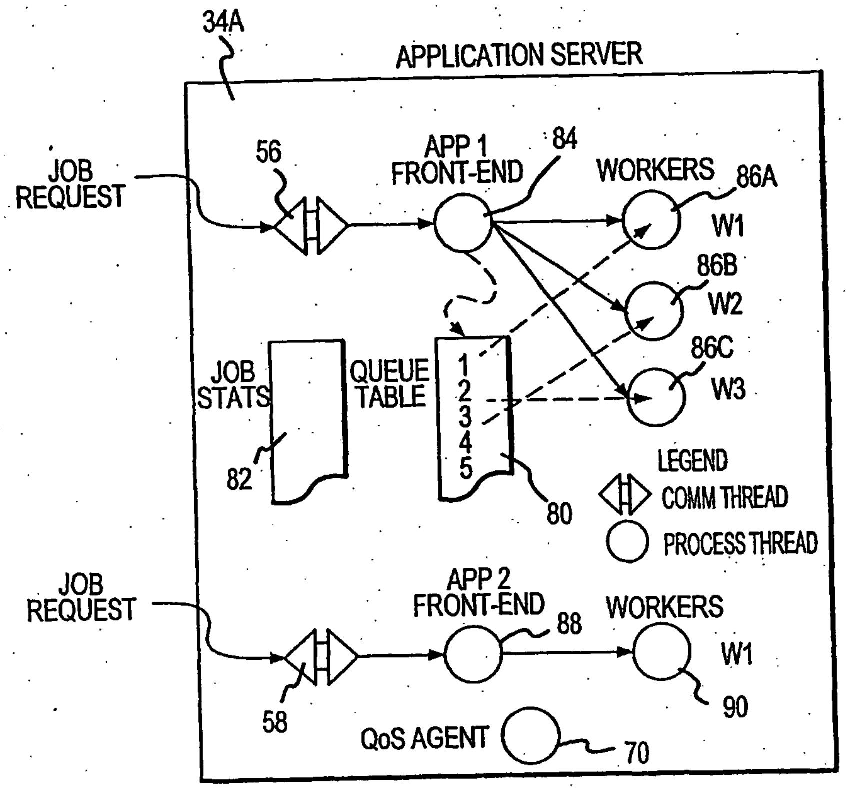

34A
APPLICATION SERVER
JOB REQUEST
56
APP 1 FRONT-END
84
WORKERS
86A
W1
86B
W2
86C
W3
JOB STATS
QUEUE TABLE
1
2
3
4
5
82
80
LEGEND
COMM THREAD
PROCESS THREAD
JOB REQUEST
APP 2 FRONT-END
88
WORKERS
W1
90
58
QoS AGENT
70

FIG. 3

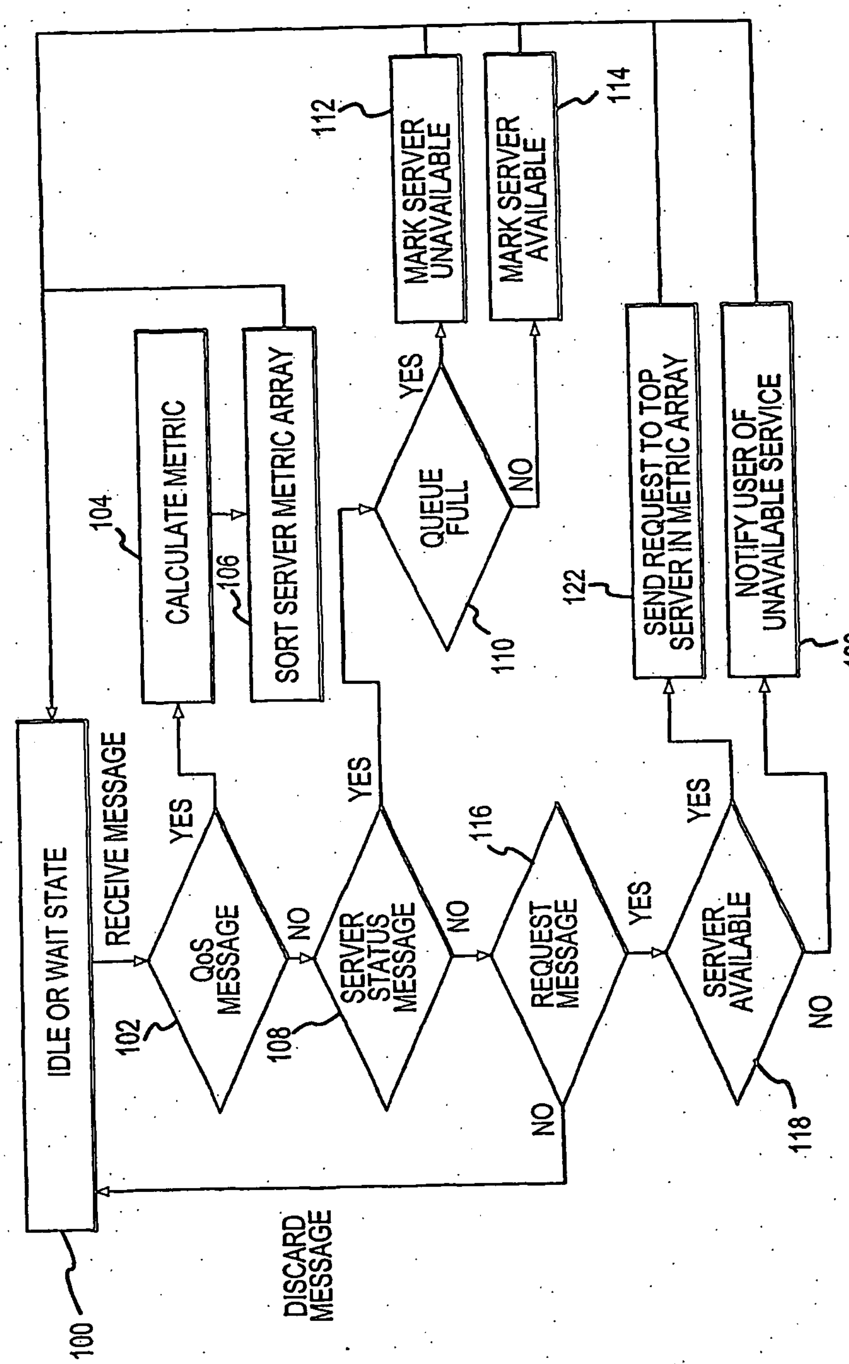

IDLE OR WAIT STATE
100
RECEIVE MESSAGE
QoS MESSAGE
102
YES
NO
CALCULATE METRIC
104
SORT SERVER METRIC ARRAY
106
SERVER STATUS MESSAGE
108
YES
NO
QUEUE FULL
110
YES
NO
MARK SERVER UNAVAILABLE
112
MARK SERVER AVAILABLE
114
REQUEST MESSAGE
116
YES
NO
DISCARD MESSAGE
SERVER AVAILABLE
118
YES
NO
SEND REQUEST TO TOP SERVER IN METRIC ARRAY
122
NOTIFY USER OF UNAVAILABLE SERVICE
120

FIG. 4

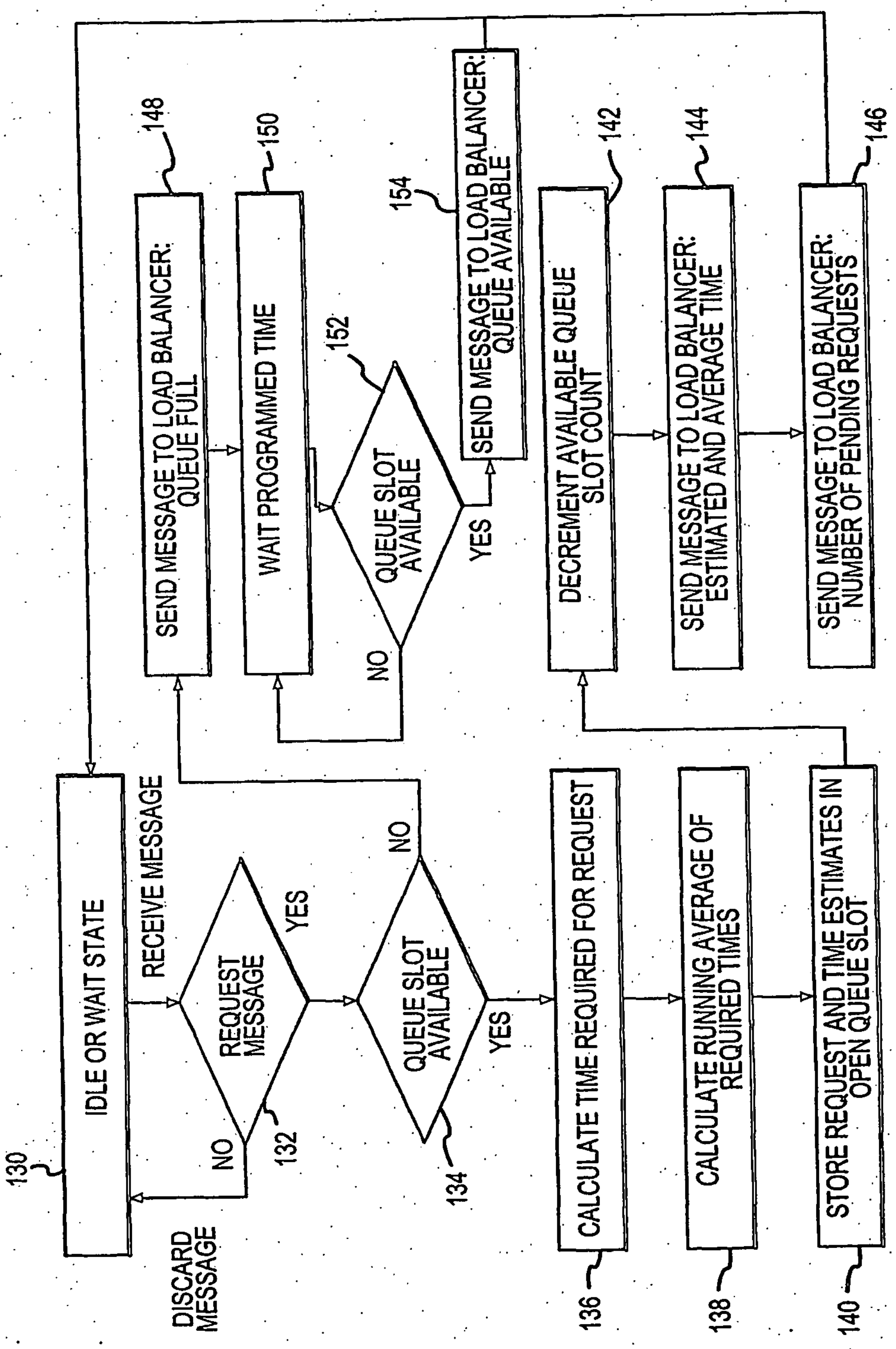

IDLE OR WAIT STATE
130
RECEIVE MESSAGE
REQUEST MESSAGE
132
YES
NO
DISCARD MESSAGE
QUEUE SLOT AVAILABLE
134
YES
NO
CALCULATE TIME REQUIRED FOR REQUEST
136
CALCULATE RUNNING AVERAGE OF REQUIRED TIMES
138
STORE REQUEST AND TIME ESTIMATES IN OPEN QUEUE SLOT
140
SEND MESSAGE TO LOAD BALANCER: QUEUE FULL
148
WAIT PROGRAMMED TIME
150
QUEUE SLOT AVAILABLE
152
NO
YES
SEND MESSAGE TO LOAD BALANCER: QUEUE AVAILABLE
154
DECREMENT AVAILABLE QUEUE SLOT COUNT
142
SEND MESSAGE TO LOAD BALANCER: ESTIMATED AND AVERAGE TIME
144
SEND MESSAGE TO LOAD BALANCER: NUMBER OF PENDING REQUESTS
146

FIG. 5

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US01/16658

A. CLASSIFICATION OF SUBJECT MATTER

IPC(7) : G06F 13/00
US CL : 709/226, 223, 224, 105
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 709/226, 223, 224, 105

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

west

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A,P	US 6,021,439 A (TUREK et al) 01 February 2000	1-21
A,P	US 6,023,722 A (COLYER) 08 February 2000	1-21
A,P	US 6,092,178 A (JINDAL et al) 18 July 2000	1-21
A,P	US 6,128,279 A (O'NEIL et al) 03 October 2000	1-21
A	US 5,774,660 A (BRENDEL et al) 30 June 1998	1-21

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

*	Special categories of cited documents:	"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A"	document defining the general state of the art which is not considered to be of particular relevance	"X"	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"E"	earlier document published on or after the international filing date		
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"Y"	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"O"	document referring to an oral disclosure, use, exhibition or other means		
"P"	document published prior to the international filing date but later than the priority date claimed	"&"	document member of the same patent family

Date of the actual completion of the international search 13 JULY 2001	Date of mailing of the international search report 14 AUG 2001
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703) 305-3230	Authorized officer Peggy Harrod GLENTON BURGESS Telephone No. (703) 305-4792